GitHub Username: raffaelcavaliere

Setlists

Description

Setlists helps you organize your music charts and makes it easy to share them with the members of your band.
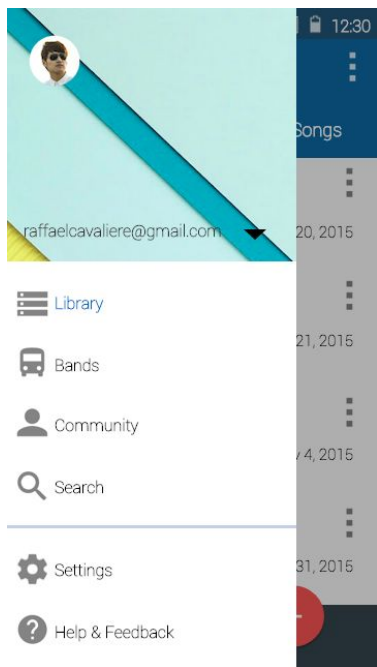
Intended User

Amateur and professional musicians.

Features

- Organize songs by artist (with Spotify search)
- Add and edit songs characteristics (tempo, duration, key, etc.)
- Attach multiple documents (versions) of a song
- Supports PDF, text, image (JPEG, BMP, GIF and PNG) and chordpro formats
- Edit text and chordpro files
- Import and export documents
- Create and manage setlists (attach a visual metronome, reorder songs, etc.)
- Send MIDI messages to instruments through android midi library
- Organize bands
- Communicate with band members
- Community features (connect to other users, download their contributions)
- Authenticate through Google Sign-in and/or Firebase authentication
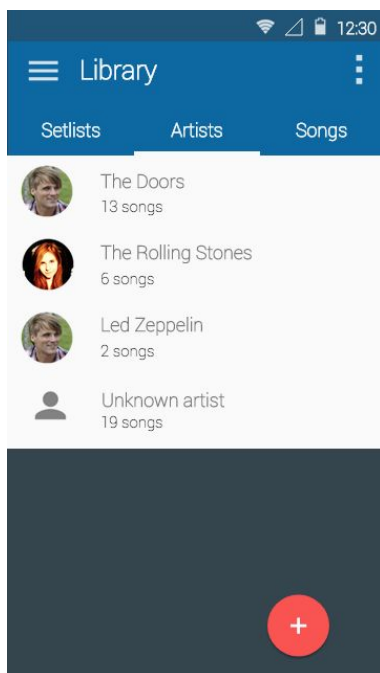
- Synchronize data between multiple devices through cloud-based repository
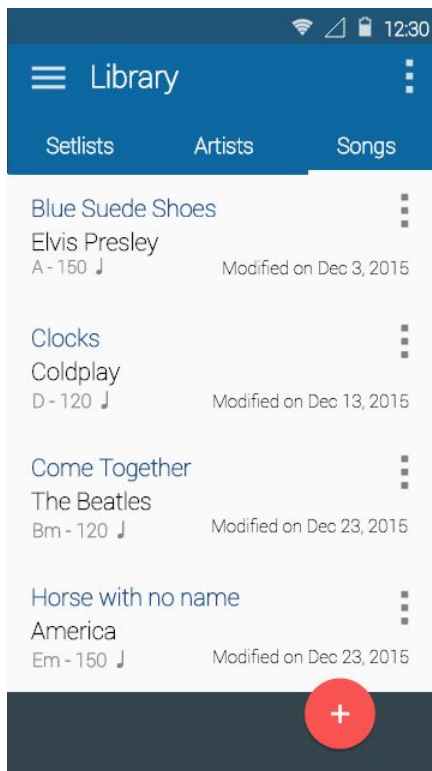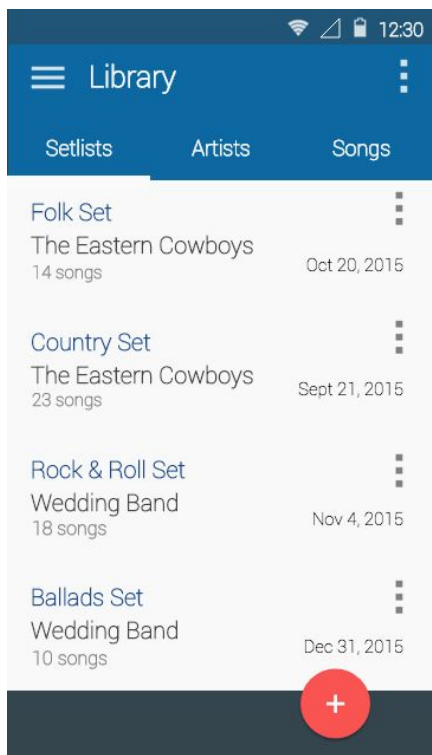
User Interface Mocks

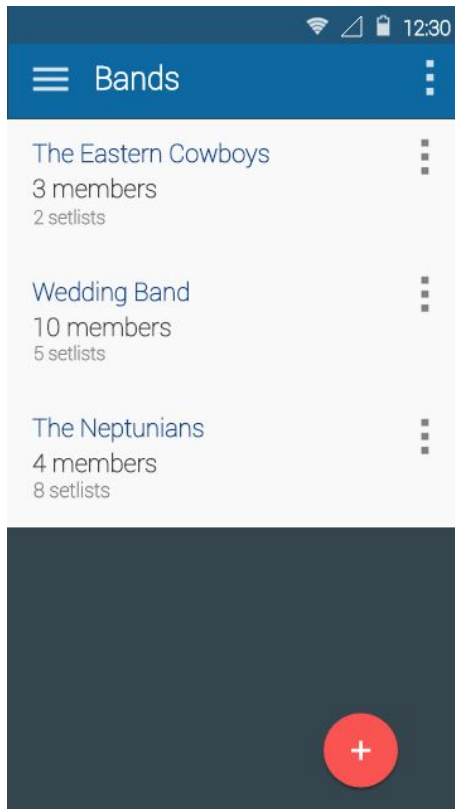## 1. Drawer menu

2. Library fragment - Artists tab

3. Library fragment - Songs tab
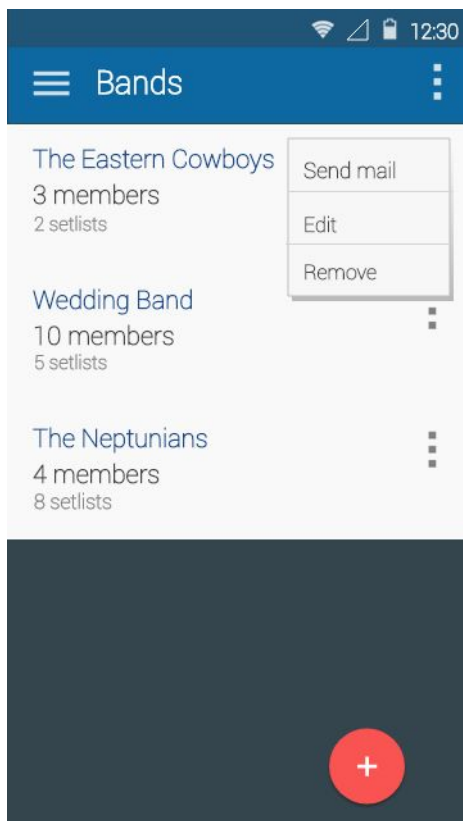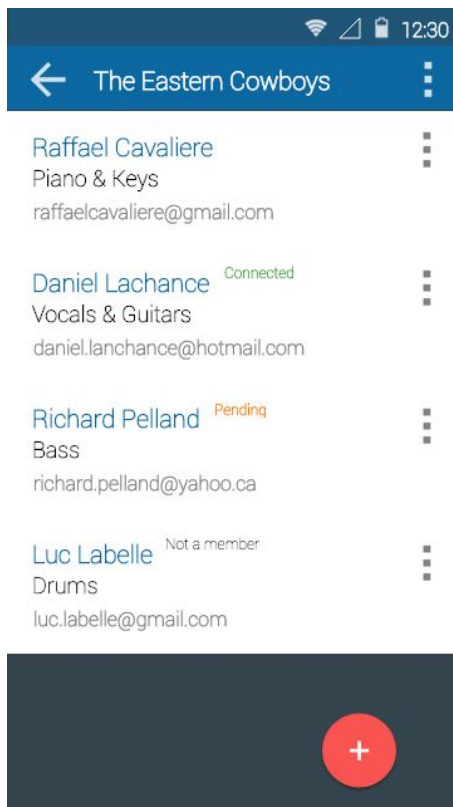

4. Library fragment - Setlists tab
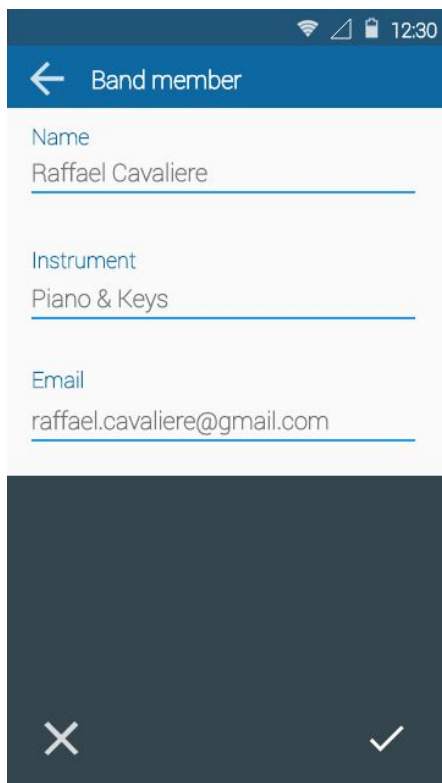
5. Bands fragment
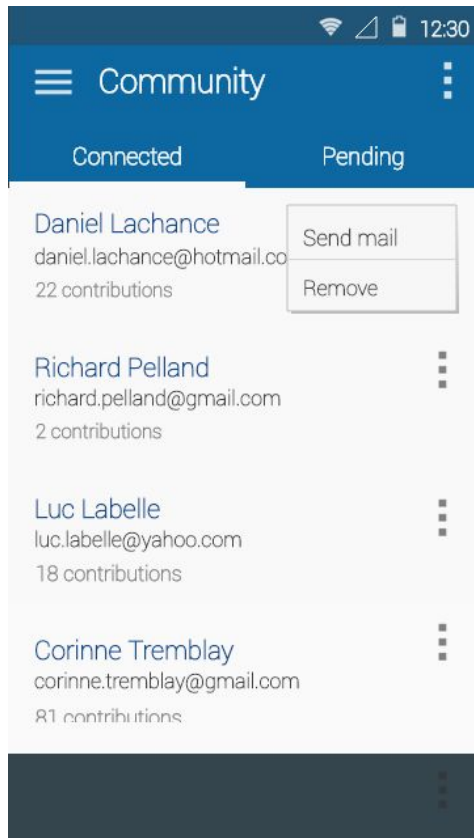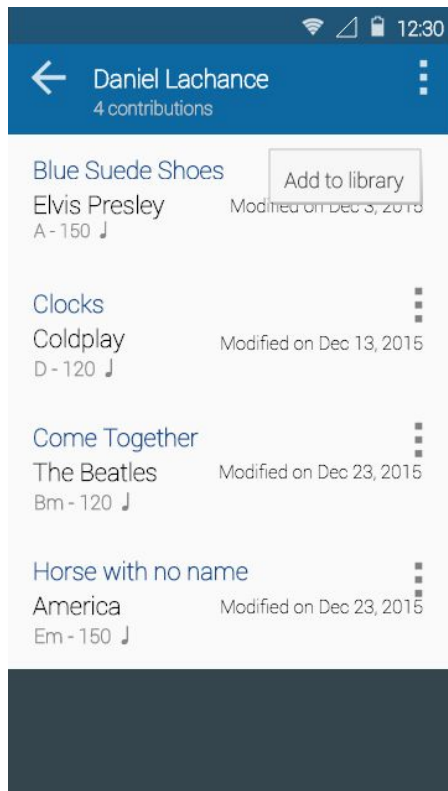

6. Band context menu

7. Band edit

8. Band member edit

9. Community fragment - Connected users

10. Community fragment - View and download songs
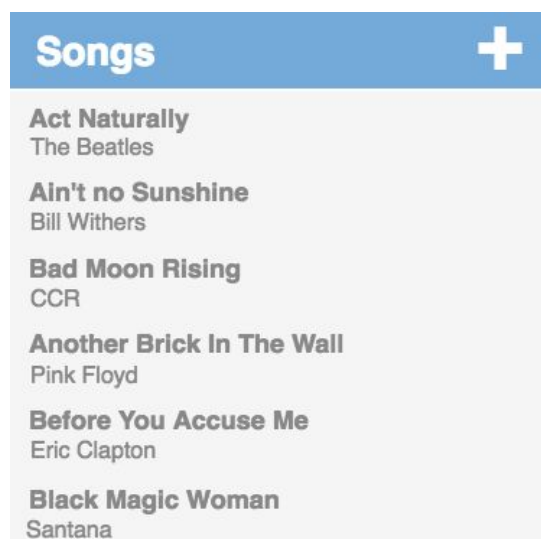
11. Song (chordpro format)

12. Widget

Key Considerations

How will your app handle data persistence?

Most of the data will be stored in a SQLite database and delivered to the application through a content provider. The documents will be saved as files in the application internal storage.

Describe any edge or corner cases in the UX.

One important thing about this application is that it's have to be fast and reliable. Since its purpose is to be used in live situations for the only benefit of the musician (who wants to have his chart up and MIDI messages sent immediately), having fancy animations between views are not necessary and not planned to be implemented.

Describe any libraries you'll be using and share your reasoning for including them.

com.google.firebase:firebase-auth - For authentication (email, google, maybe others)
com.squareup.picasso:picasso - For image caching
com.github.kaaes:spotify-web-api-android - For spotify artist/song search
com.github.barteksc:android-pdf-viewer - For viewing PDFs
Com.google.firebase:firebase-storage - For storing documents (versions)

Out of the box android.midi library !

Describe how you will implement Google Play Services or other external services.

Google sign-in will be used, although it must be determined if the new Firebase platform is sufficient and maybe a better option (or a combination of both).

External service (repository and Rest API) will either be developed in Python (Django) or C# (WebAPI) and hosted on Azure.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Find, test and configure libraries for :

- ○ Search for an artist picture
- ○ Authenticate with a google account
- ○ Parse chordpro documents
- ○ Send MIDI messages
- ○ File pickering
- ○ Drag and drop
- ○ Display PDFs
- Determine a project structure (activities, fragments, layouts, menu, etc.)
- Upgrade to the latest Android Studio, gradle, AppCompat versions, etc.

Task 2: Create database objects

- Create database contract
- Create database
  - ○ Create artist table
  - ○ Create band table
  - ○ Create document table
  - ○ Create musician table
  - ○ Create song table
  - ○ Create setlist table
  - ○ Create setlist_song table
  - ○ Create message table
  - ○ Create document_message table
- Create cursor loaders
- Create content provider
- Create parcelables

Task 3: Create Main activity objects

- Create drawer layout and menu
- Create main activity layout and AppCompatActivity

Task 4: Create Library fragment objects

- Create Library layout and Fragment (with TabLayout)
- Create Artists tab layout and Fragment
- Create Add/Edit Artist layout and Activity
- Implement Spotify search for the artist picture
- Create Songs tab layout and Fragment
- Create Add/Edit Song layout and Activity
- Create Document Add/Edit layout and Activity
- Create FilePicker layout and Activity for choosing document
- Create Document Viewer Activity (for Text, PDF, Image or Chordpro formats)

- Create Sets tab layout and Fragment
- Create Add/Edit Set layout and Activity
- Create Set details layout and Activity (with reorder)
- Create SongPicker layout and Activity for choosing songs (multi-select)
- Create Multiple Document Viewer Activity (with swiping)
- Create context menus and toolbar menus for all activities and fragments
- Implement Song editing (for text and chordpro formats)
- Implement Document Export

Task 5: Create Band fragment objects

- Create Band layout and Fragment
- Create Add/Edit band layout and Activity
- Create Members layout and Activity
- Create Add/Edit Member layout and Activity
- Implement send email to Band, Member
- Create context menus and toolbar menus for all activities and fragments

Task 6: Create MIDI fragment objects

- Create Midi layout and Fragment
- Create Add/Edit MIDI message layout and Activity
- Implement send MIDI message
- Create context menus and toolbar menus for all activities and fragments

Task 7: Create back-end database

- Create database
  - Create user table
  - Create artist table
  - Create band table
  - Create document table
  - Create musician table
  - Create song table
  - Create setlist table
  - Create setlist_song table
  - Create message table
  - Create document_message table
- Create UnitOfWork and database context (.NET)

Task 8: Create RESTFul services

- Implement CRUD operations on all tables

- Implement OAuth2 authentication
- Provide a basic out of the box UI for executing requests online

Task 9: Implement synchronization

- Implement synchronization through an IntentService
- Synchronize on demand and on application start
- Push changes as they occur
- Implement simple replace instead of merge, based on datetime stamping

Task 10: Implement cloud storage

- Set up Firebase Cloud Storage
- Implement upload
- Implement synchronization for documents

Task 11: Create Community fragment objects

- Create Community layout and Fragment (with tabLayout)
- Create Connected tab layout and Fragment
- Create Requests tab layout and Fragment
- Create context menus and toolbar menus for all activities and fragments

Task 12: Create Search activity objects

- Create Search layout and Fragment (with tabLayout)
- Create Search for People tab layout and Fragment
- Create Search for Song tab layout and Fragment
- Create context menus and toolbar menus for all activities and fragments

Task 13: Create Settings activity objects

- Create Settings layout and Activity
- Settings
  - Theme (Material Dark / Material Light)
  - Backup
  - Import batch
  - Restore
  - Always on
  - Backlight
  - Etc.

Task 14: Implement login and authentication

- Implement Firebase Authentication
- Implement Logout

Task 15: Create widget

- Create widget layout and AppWidgetProvider
- Connect widget to song viewer and song Add functions

Submission Instructions
- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:
- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"